Victorian 6502 User Group Newsletter

KAOS

For People Who Have Got Smart

OSI SYM KIM AIM UK101 RABBLE 65

Registered by Australia Post
Publication No. VBG4212

ADDRESS ALL CORRESPONDENCE TO 10 FORBES ST., ESSENDON, VIC. 3040

Vol.5 Nos.3 & 4 December 1984 & January 1985

If you are still chasing the postman for your December KAOS, you can stop, there wasn't one! Due to a couple of unfortunate occurrences (the printer closing early for Christmas and Ian having a spell in hospital), for the first time in over four years we missed an issue. To make amends we have increased this issue to 2Ø pages, and given you two Forth articles and two Superboards.

A reminder that memberships are now due, if we have not received your renewal by the 14th February you will not receive the next newsletter. Fees are still $15, $2Ø and $25 for Australia, New Zealand and PNG and elsewhere respectively.

George informs us that as there has been so much trouble with the answering machine on Compsoft's phone, that he has decided to do business through a postal address, which is Mr G. Nikolaidis,
. George will still be supporting CP/M but he has given KAOS the rights to CompDOS and Dabug. More about this next month.

At the January meeting, George will be bringing along for sale stock that remained after Compsoft closed. Some items will be sold at unbelievably low prices!!

The next meeting will be on the 27th January 1985 at 2pm at the Essendon Primary School which is on the corner of Raleigh and Nicholson Streets, Essendon. We realise that this is a holiday weekend and some members may have other plans but we will have a few interstate members visiting and it would be nice for them to have a lot of friendly faces to talk to. The school will be open at 1pm.

The closing date for articles for the next newsletter will be 8th February.

INDEX

ROM TEST EXPANSION
by John Whitehead

This is my attempt at expanding the ROM Integrity Test program in the October KAOS.

It automatically tests each BASIC ROM in turn by checking each byte in a ROM and adding them all together but discarding overflow. It prints an error message and the checksum found if a ROM is faulty.

To test that the 'fault printing code' works, change a byte value in 'SUM2'.

The program is 'run' at 'start' and contains no self modifying code, so can be put in EPROM. The scratch pad is in page zero at $ØØ4Ø.

My value for BASIC 4 is different due to it being an EPROM with garbage bug fixed.

I challenge you budding machine code programmers with your 8K cassette Superboards or your 32K disk machines to get out your Assembler and try to improve my code by making it shorter or better. The best will be published in KAOS. Send your well commented source code on cassette or paper to me at

```
1Ø   ØØØØ              SUMF=$4Ø                  ;CHECKSUM FOUND
2Ø   ØØØØ              SUME=SUMF+1               ;CHECKSUM EXPECTED
3Ø   ØØØØ              ROM=SUMF+2                ;ROM BEING INSPECTED
4Ø   ØØØØ              ADD1=SUMF+3               ;ADDRESS INSPECTED
5Ø   ØØØØ              SCREEN=$FFEE
6Ø   ØØØØ              ;
7Ø   1FØØ              *=$1FØØ
8Ø   1FØØ AØØ1         START    LDY #1
9Ø   1FØ2 8442                  STY ROM          ;SET FOR BASIC 1
1ØØ  1FØ4 A442         NEXT     LDY ROM
11Ø  1FØ6 B9941F                LDA ADD2,Y       ;GET ADD HI
12Ø  1FØ9 8544                  STA ADD1+1
13Ø  1FØB B9951F                LDA ADD2+1,Y     GET ADD HI END
14Ø  1FØE 8545                  STA ADD1+2
15Ø  1F1Ø B99A1F                LDA SUM2,Y       ;GET SUM EXPECTED
16Ø  1F13 8541                  STA SUME
17Ø  1F15 AØØØ                  LDA #Ø
18Ø  1F17 8443                  STY ADD1         ;SET ADD LO TO Ø
19Ø  1F19 98                    TYA              ;SET ACC TO Ø
2ØØ  1F1A 18                    CLC
21Ø  1F1B 7143         MORE     ADC (ADD1),Y     ;GET BYTE VALUE FROM ROM
22Ø  1F1D E643                  INC ADD1
23Ø  1F1F DØØ2                  BNE PASS
24Ø  1F21 E644                  INC ADD1+1
25Ø  1F23 A643         PASS     LDX ADD1
26Ø  1F25 EØØØ                  CPX #Ø           ;IS ADD LO = Ø
27Ø  1F27 DØF2                  BNE MORE
28Ø  1F29 A644                  LDX ADD1+1
29Ø  1F2B E445                  CPX ADD1+2       ;IS ADD HI AT MAX.
3ØØ  1F2D DØEC                  BNE MORE
```

```
310 1F2F 8540              STA SUMF        ;STORE THE TOTAL FOUND
320 1F31 C541              CMP SUME        ;TEST RESULT
330 1F33 D00B              BNE FAULT
340 1F35 E642              INC ROM
350 1F37 A542              LDA ROM
360 1F39 C905              CMP #5          ;IS IT THE END
370 1F3B D0C7              BNE NEXT
380 1F3D 4C00E8   END      JUMP $E800      ;TEST OVER. GOTO EXMON
390 1F40                                   ;
400 1F40 A000     FAULT    LDY #0          ;SET CHAR. POINTER TO ZERO
410 1F42 B9781F   AGAIN    LDA TEXT,Y      ;GET A CHAR.
420 1F45 C900              CMP #0          ;IS IT THE LAST CHAR.
430 1F47 F00E              BEQ CHECK
440 1F49 C901              CMP #1
450 1F4B D004              BNE OVER
460 1F4D A542              LDA ROM         ;GET ROM #
470 1F4F 0930              ORA #$30        ;CONVERT TO ASCII
480 1F51 20EEFF   OVER     JSR SCREEN      ;PUT CHAR. ON SCREEN
490 1F54 C8                INY             ;INC. CHAR. POINTER
500 1F55 D0EB              BNE AGAIN       ;ALWAYS
510 1F57                                   ;
520 1F57 A540     CHECK    LDA SUMF        ;CONVERT 1 BYTE TO 2 ASCII CHAR.
530 1F59 29F0              AND #$F0        ;GET 4 M.S. BITS
540 1F5B 4A                LSR A
550 1F5C 4A                LSR A
560 1F5D 4A                LSR A
570 1F5E 4A                LSR A
580 1F5F 206B1F            JSR ASCII
590 1F62 A540              LDA SUMF
600 1F64 290F              AND #$0F        ;GET 4 L.S. BITS
610 1F66 206B1F            JSR ASCII
620 1F69 D0D2              BNE END         ;ALWAYS
630 1F6B                                   ;
640 1F6B          ASCII    CLC             ;CONVERT HEX TO ASCII
650 1F63 C90A              CMP #$A
660 1F6E 9002              BCC LESS        ;IF IT'S > 9, ADD 6
670 1F70 6906              ADC #6
680 1F72 6930     LESS     ADC #$30
690 1F74 20EEFF            JSR SCREEN
700 1F77 60                RTS
710 1F78                                   ;
720 1F78 20       TEXT .BYT ' BASIC ',1,' FAULT. CHECKSUM = ',0
720 1F79 42 61 73 69 63 20 01 20 66 61 75 6C 74
720 1F86 2E 20 43 68 65 63 6B 73 75 6D 20 3D 20 00
730 1F94 00      ADD2 .BYT 0,$A0,$A8,$B0,$B8,$C0
730 1F95 A0 A8 B0 B8 C0
740 1F9A 00      SUM2 .BYT 0,$6B,$3D,$1D,$79
740 1F9B 6B 3D 1D 79
750 1F9F                   ;4TH BYTE CHANGED FOR BASIC 3 GARBAGE BUG
760 1F9F                                   ;
770 1F9F                                   ;...... THE END ......
```

SUPERBOARD

© December, 1984
Newsletter of the Ohio Superboard User Group, 146 York St, Nundah, 4012.

I regret that I cannot print the promised adventure game this month. The main reason for this is that I wanted it coded in generic form, as KAOS members have various computer types. Some persistent bugs remain. For those who want an adventure to play for christmas, I can recommend the one from the last OSUG programming competition, entered by Sean Davidson.
A slightly modified version of this is in the KAOS library at a cost of $3. Contact John Whitehead for more details.

Alternately I can recommend the Victory Package, reviewed in KAOS 3/6 and 3/8. This consists of lots of excellent arcade games and also adventures. Victory Software wrote to me a few weeks ago with the news that they are closing out their OSI inventory. This is sad news. However they now offer both Victory Packages (3 tapes) at the bargain basement price of US$15, which includes airmail postage. Address is : VICTORY SOFTWARE CORP, 1410 RUSSELL ROAD, PAOLI, PA 19301, U.S.A.

(OSI Greatest Hits Vol 1 is available separately at US$10)

SOFTWARE REVIEW - *Small Business Analysis.*

This program was written by a Dr. Owen, and originally sold by Aardvark. The program was written to fit into a small computer with as little as 4k of memory, "currently costing about $1000 ". (In 1979 that is)
So the program is written as efficiently as possible as regards memory usage, and you enter the data as data statements and save it with the program.
There are six analytical methods built in. These are:-

(1) Cash flow analysis.
(2) Current financial condition.
(3) Operating (income) statement analysis.
(4) Break even analysis.
(5) Standard financial ratios.
(6) An econometric model of the firm.

Essentially, the program is an elementary spreadsheet, with twenty cells. It also has predictive capability, according to the instructions.

The program comes with a very comprehensive manual of some 14 pages. All the analytical methods are fully explained, with the help of illustrations, and a hypothetical example of a small business is provided also.
The documentation says that up to 12 years of data can be stored in the program without exceeding memory capacity.

Not having a great knowledge of business operations, I find it difficult to comment on the usefulness of this U.S. program when applied to Australian conditions. However enough information is provided to enable modification of the program to suit. This is one of the best documented programs I've seen, and is therefore worth a look if you are interested in these analyses. It should run on any OSI machine.

Small Business Analysis is available in the OSUG library for library members for one 60¢ and one 30¢ stamp to cover postage costs.

NEXT MONTH

Hopefully, the Adventure promised this month. A review of some educational software from the U.K. IBM clones - what they offer and the cost.

— SUPERBOARD —

An efficient mailing list program for O.S.I.

INITIAL CONSIDERATIONS.

KAOS, and most other publishers use the Registered Publications postal service. This is truly an excellent service for clubs, church groups and etc, where there is a regular mailing. There is a substantial discount on the regular mailing service, however there are several limitations.

Items always travel surface mail, and have to be pre-sorted by the sender according to a rather complex and sometimes incomprehensible set of rules. There are of course, myriads of rules and regulations in typical bureaucratic style, along with the usual dire threats should these rules and regulations be infringed by the user, even if accidentally, or because the aforesaid rules and regulations sometimes contradict.

For about a year before merging the OSUG newsletter with KAOS, I used the service. The method was very simple, as labels on A4 size sheets are available and I have access to a photocopier. Sorting was always an annoyance, even with just over 100 members. Everything was done by hand, of course. At the start, I typed all the members names and addresses in postcode order, but as some moved to other areas, and new members joined, the list got more and more erratic. At the end it was a hopeless mess, with nearly 15% of the labels being wasted.

Now KAOS store the membership details on a system with twin 8" disks, as revealed in the May,1983 article, and the storage requirement was over 200k. Plainly, this amount of data storage is way beyond a cassette based system with 8 or 16k of memory. However if you restrict your program to just a mailing list capability, a lot can be done.

For programming purposes, it is important that labels be in grouped, postcode order. There are some 80 mail centre groupings, and postcodes in rather muddled orders can go to the one centre, so for maximum efficiency, it will be necessary to do some pre-sorting of names and addresses <u>before</u> typing in this data. Otherwise you will have to do it by hand for every mailing, which is tedious, believe me! Also when entering new data for new members, or address changes, the order will need to be preserved. (Remember, we don't have the memory space to allow ourselves the luxury of a sort routine).

PRINTER BLUES.

You will need a tractor feed printer and label paper. The cost is just over a cent a label. If you use envelopes, I would recommend buying them from a printer, already printed with the inscription and postal marks which are required by Australia Post. These work out at about 6¢ a pop. If the publication is A4 size sheets, get end-opening envelopes size 9" x 4" so that a minimum of folding and inserting effort is required. The items sent by the Registered Publications service must be all the same, and the envelope cannot be sealed against inspection, hence the end-opening flap, easy to tuck inside. Saves a gluey tongue too!
Mailing labels can come 3 or 4 wide, and be various sizes. You can also buy them one wide. This latter makes for the easiest mailing program imaginable. Simply store your details as WP-6502 files. The word processor makes for simple insertion or deletion of entries, and moving records about to suit the changing whims of the postal service is made gloriously simple. If at all possible, the one-wide label method should be used. Big savings in typing can be made by judicious use of the global edit capability of WP-6502.

— SUPERBOARD —

MORE THOUGHTS

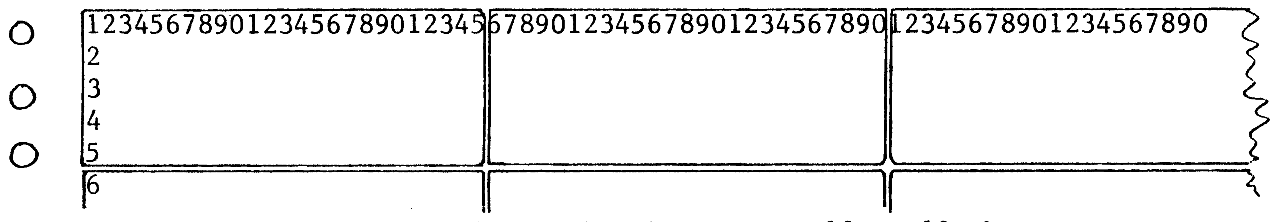
1000 DATA John Citizen,

Well, that's the sort of data we need, 5 fields, covers everything. OK? Well it could be, but the line takes up nearly 50 bytes of memory, and we're a bit short on storage. What about some shortcuts? John Citizen could become J.Citizen. That only saves 4 bytes but then there will be the odd Percival and even longer names sometimes inflicted by parents. Even 4 x 300 names equals 1200 bytes saved in typing and memory.
How about putting S for street and R for road, and then getting the program to print out the whole word? Not on, I'm afraid! People live in Alleys, Avenues, Crests, Courts, Crescents, Drives, Highways, Parades, Places, Rises, Terraces, and Ways - to name some of the more common ones. To find out the abbreviations the experts use, consult the pages of the telephone directory. Telecom has determined that most people live in streets, and ignore that word entirely. All the other approved abbreviations, Crt, Crs, Tce and etcetera will be found within that tome, and will save considerable space. You could use a * character for Street, and insert it from the driver routine, but might as well just use the abbreviation, St.

Finally, there's the NSW. Australia Post doesn't need it! The first figure of the Postcode is the State Key. It may interest residents of the ACT that they are part of NSW. NT is also considered part of SA. Why the figures 1, 8 and 9 were not used is anybody's guess. So now we have:-

1000 DATA J.Citizen, (9 bytes shorter)

Most of the hard work has now been done. Having typed in your data in grouped postcode order, with line spacing of 10 to allow for further entries, it is now time to consider the main program.
Let us assume your labels come three-wide. Some experimentation now needs to be done to find out the details of the tabbing requirements.



You may decide to use double height characters, 12 or 10 characters per inch horizontally, and 6 or 8 lines per inch vertically depending on your printer's capabilities. Whatever, now is the time to experiment and decide.

```
  1Ø PRINT"Cassette Mailer #1":PRINT
  2Ø M=3:T=25:INPUT"Ready to print";N1$
  3Ø POKE 517,1
  4Ø READ N1$,A1$,C1$,N2$,A2$,C2$,N3$,A3$,C3$
  5Ø PRINT:PRINT TAB(M);N1$;TAB(M+T);N2$;TAB(M+T*2);N3$
  6Ø PRINT TAB(M);A1$;TAB(M+T);A2$;TAB(M+T*2);A3$
  7Ø PRINT TAB(M);C1$;TAB(M+T);C2$;TAB(M+T*2);C3$
  8Ø PRINT:IF N3$="END" THEN POKE 517,Ø:END
  9Ø GOTO 4Ø
```

Your last DATA line should read: 1ØØØØ DATA END,1,1,END,1,1,END,1,1
else the program might fail to print the last entry. Other lines should print the Area Bundle labels needed for the Registered Post service at the appropriate places to make final sorting a thing of the past. If you need more than one program to store your whole mailing list, just duplicate lines 1Ø to 9Ø in each one. The second one would be "Cassette Mailer #2".

Ed Richardson.

UNDERSTANDING FORTH, THE COMPILER, by Ray.Gardiner

The forth machine can operate in two distinct states, the first of these is the INTERPRET state, where all input to the text interpreter is executed if the input is executable. The second state is the COMPILE state where all input is compiled into the dictionary, rather than interpreted.

The words we will be using to control the state of the forth machine and the actions of the compiler are as follows.

word	description / comments
[	switch to interpret state.
]	switch to compile state.
immediate	mark the word last defined as IMMEDIATE, ie executes while the system is in the compile state.
[compile]	force the compilation of the following immediate word.
compile	compile the word following at execution time.
state	system variable containing the compilation state, if the state is non zero then the system is in the compile state.

Unless you are some sort of genius the above definitions will mean absolutely nothing to you, so stay tuned, here is the explanation.

Suppose you wished to stop compiling temporarily, say to calculate a numeric value or perhaps some debugging commands are required. Or perhaps change the I/O base, for example :

```
: TAP-ON  [ BINARY ] 10000000  [ HEX ] C721 ! [ DECIMAL ] ;
```

In the above example we are simply switching the compiler on and off so that we can change the I/O base purely for convenience, consider what would happen if the compiler wasn't switched off, instead of changing the base the word BINARY would have been compiled rather than being executed.

Now for some interesting definitions, namely the [and] themselves.

```
: [  0 state ! ; IMMEDIATE     ( stop compiling          )
: ] C0 state ! ;               ( enter compilation state )
```

You will notice that the word to stop compiling, is marked as immediate the reason for this is that it must EXECUTE rather than compile even when the system is in the compile state. Now that everyone is thoroughly confused, maybe it would be a wise move to try some simple examples.

IMMEDIATE and what it means.

IMMEDIATE indicates just what it says, any word marked as immediate will execute whenever it is encountered irrespective of whether the system is in compile state or not.

```
: TEXT1 ." this will be non-immediate i.e. normal? "  ;
: TEXT2 TEXT1 ;
```

When TEXT1 is typed in the interpret state the sentence will be output to the video display, so far all is normal. Now what would have happened if TEXT1 had been immediate?

```
      : TEXT1 ." this will be the IMMEDIATE version " ; IMMEDIATE
      : TEXT2 TEXT1 ;
```

Now, what happens is that the text is output during the compilation of TEXT2.

Trivial examples such as these are only a very shallow indication of the power that can be exercised over the compilation process by using IMMEDIATE words to control and modify the operation of the compiler while it is in full flight.

Consider the DO..LOOP structure that we described earlier in this series, you could write your own looping primitives. I have omitted the error and syntax checking for simplicity.

```
      : DO   COMPILE (DO)   HERE     ; IMMEDIATE
      : LOOP COMPILE (LOOP) HERE - , ; IMMEDIATE
```

Both of these words execute when the system is in the compile state, the DO leaves the dictionary pointer indicating the address to be used as the target of the backward branch. LOOP simply subtracts the addresses and then comma's the difference to be used later by (loop) at execution time.

Problem?, what if I wish to change the word DO to something else, or maybe embed the definitions of DO and LOOP, (or any other immediate word for that matter) inside a new definition.

```
      : FOR  DO ;
      : NEXT LOOP ;
```

This approach will fall in a heap with unfriendly messages about your ancestory and general personal abuse, what happens is that DO and LOOP are immediate and as such will try to execute rather than be compiled. The solution is to use [COMPILE].

```
      : FOR [COMPILE] DO ;
      : NEXT [COMPILE] LOOP ;
```

The [COMPILE] scans ahead in the input stream and compiles the next word immediate or not.

```
      : YECCHS  0   FOR  CR ." Well it looks BASIC enough "  NEXT  ;
```

For those with an insatiable thirst for knowledge, here is the definition of [COMPILE] and IMMEDIATE.

```
      : [COMPILE]  -FIND 0  0 ?ERROR DROP CFA , ; IMMEDIATE

      :  IMMEDIATE  LATEST 40 TOGGLE ;  ( toggle the precedence bit in the NFA
                                          of the last word defined )
```

Refer to last month's discussion on dictionary structure. for CFA,NFA and LATEST.

We have looked at various ways of controlling the compiler, but completely ignored the actual mechanics of the compiler operation.

In the body of a definition the compiler simply scans the input stream and looks up each word in the dictionary and then comma's the CFA into the dictionary. At the start of a new word we have to do a bit more work to build the name, and link fields and check to see if any other words of the same name were previously defined. The word is CREATE, which builds the header and checks for duplicated names.

```
: CREATE
          -FIND                    ( search dictionary for duplicate names )
              IF DROP NFA ID.  ( not unique, so warn user )
                 4 MESSAGE SPACE
              THEN
              HERE DUP C@ WIDTH @ MIN 1+ ALLOT
              DUP  A0  TOGGLE HERE 1 -  80 TOGGLE
              LATEST ,          ( link to last word )
              CURRENT @ !       ( update vocab pointer )
              HERE 2+ ,    ;    ( code defn rewritten later )
```

CREATE is used in the definition of : and CONSTANT , in each name field the header byte contains a bit which is tested during the dictionary search to determine if a word is available for execution or compiling. This bit is called the SMUDGE bit for reasons that will become obvious. Before any new definition can be executed or compiled it must be UNSMUDGED. In a normal : definition the word isn't made available until the ; is encountered, however if you wish to write recursive code you may choose to unsmudge the word being compiled. There are neater ways of writing recursive code that we will discuss in a future article.

```
: SMUDGE  LATEST 20 TOGGLE ;

: :  ?EXEC !CSP                ( error checking )
     CURRENT @ CONTEXT !       ( vocabulary      )
     CREATE ]                  ( build header and turn compiler on )
     ;CODE    IMMEDIATE        ( rewrite code field of word being
                                 defined to point to DOCOL code )

     DOCOL                     ( machine language for DOCOL follows)
```

Notice that : doesn't terminate in a ; as the other colon definitions we have examined have done. The terminating word is ;CODE which we will discuss in greater depth when we look at defining words, such as the <BUILDS DOES> structures next month.

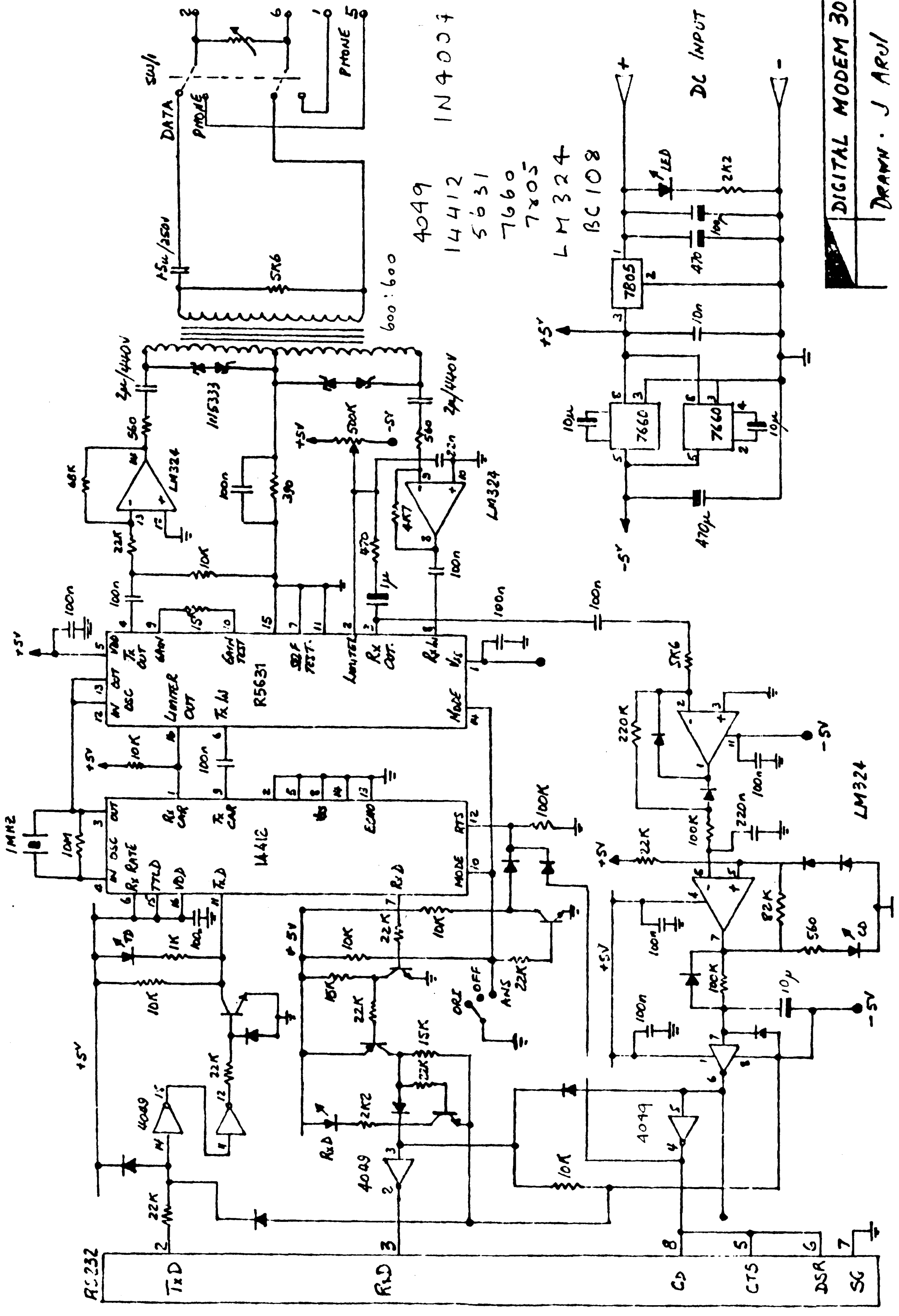
```
: XMAS NUMBER_IN_FAMILY 0 DO  BUY_PRESENT LOOP ." Merry Xmas " ;
```

D.I.Y. MODEM
designed by Julian Arul

One of our Sydney members, Norm Bate, sent us the following item.
Here is the circuit for a do-it-youself modem that was designed by KAOS member Julian Arul.

The R5631 chip costs about $18.ØØ and the 776Ø regulator about $3.45. The Zener diodes on the transformer are only for protection.

Julian has applied to Telecom for approval.



THE MAD HACKER'S TEA PARTY

This is the title of the 12 page newsletter we recently received from Eric Lindsay which outlines his plans to design and build 68000 computers.

Below are extracts from the newsletter which give some of his thoughts on the project. If you are interested in learning more send $2.50 to Eric at and he will send you a copy.

HOW DO YOU GET THE NEWSLETTER?

Show interest, by design contributions, letters or phone calls about design problems. If I don't hear from you, I'm not going to waste my money sending you a copy. If you really can't contribute, send $2.50 per issue (this price is set high enough to discourage onlookers).

OUR AIM!

To design and build a 680XX equivalent of the original Trash 80 and Rotten Apples. A machine with which hackers can have fun. This implies a totally open architecture, with full specifications of how everything works. In short, no trade secrets. We may however copyright some of the code, because it would be nice if any money made on this project were to come to the people who did the work.

DESIGN OUTLINES

We should probably try for two designs, the easy approach (code name Hacker One), and the ambitious upmarket version (code name Big Mac).

Hacker One should be a cheap development system, preferably compatible with Hal Hardenberg's Dtack Grounded co-processor for the Pet and Apple. This would give us a source of code, and save a lot of time.

Big Mac would be memory mapped in a similar manner to a certain fruity computer, but with more memory, larger capacity disk drive, provision for hard disk, and at least 8 colours on the screen, which would be of higher definition. It would probably run as slow as a wet weekend using just a 68000, as we are talking about >32K of video memory alone. We should be thinking in terms of bit mapped graphics displays of about 700+ by 500+. This implies spending at least $700 on the display alone, and as much again on the hard disk. Obiviously the fully figured conversion would be costly, however we should try to keep the direct board costs down, for those running floppy and monochrome monitors.

RABBLE 65
by Jeff Rae

I am putting this item in KAOS because I am spending an increasing amount of time on the phone answering queries from Rabble 65 owners. As most of the queries seem to relate to the same topics e.g. using the Rabble 65 monitor from a DOS, software available and what programs do, and the operation of the screen windows, it appears to me that they could more easily be answered through a column in KAOS. To get a better idea of what questions you want answered, if you contact me at the address below, preferably by writing but by phone if you must, I will either write a short article myself or persuade?? another Rabble owner to do it. If anyone has any hints or tips that they think will be of interest to other owners they can either send them to me or direct to KAOS.

My address is: Mr J. Rae,

BACKING UP APPLE II ROMS ON DISK

BY NOEL COWARD

I find it good insurance to copy all my proms, such as Eprom Burner, 80 column card and Character Generator onto disk.

If I lose one of these Proms I can, with the aid of an Eprom burner card make a replacement Prom.

As the Apple etc. has its main program in Prom (or Rom) I have also copied these programs to disk.

(a) Boot disk with DOS.

(b) Press control-reset after DOS is booted (or reset).

(c) Type CALL-151 ↵ (↵ = return. To get into monitor)

(d) Type 1ØØØ < DØØØ. FFFFM ↵ (Shifts $DØØØ - $FFFF to memory $1ØØØ-$2FFF)

(e) Type 3DØG ↵ (To get out of monitor)

(f) Type BSAVE APPLE ROMS, A $1ØØØ, L $3ØØØ ↵ (Loads program to disk)

If you need to replace one of your Proms later you can take your disk to a friend who has an Eprom burner card and manufacture a replacement from the Prom data stored on disk.

Prom Locations	DØ	$1ØØØ,	L$8ØØ	(for 2716'S)
	D8	$18ØØ,	L$8ØØ	
	EØ	$2ØØØ,	L$8ØØ	
	E8	$28ØØ,	L$8ØØ	
	EØ	$3ØØØ,	L$8ØØ	
	F8	$38ØØ,	L$8ØØ	

WARNING

In the genuine Apple II they have Roms not Eproms.
If a Rom is replaced with an Eprom modifications are needed.
To replace one of the six Roms with a programmed 2716 Eprom pin 21 which have to go to +5V not to Pin 20 and Pin 18 should be connected to GND not to the bus $\overline{\text{INH}}$ from the peripheral I/O.

All the Apple look-a-likes have Eproms, 2716's, 2732's or 2764 and a 2732.

THE CP/M FILE

We had a very nice day for the BBQ, and those that turned up enjoyed it immensely. Unfortunately, for reasons known only to those who stayed away, a lot did (stay away, that is). Let's hope that they all enjoyed whatever they were doing too.

Paul and Michael turned up in their new wheels, and a couple of high-performance bits of machinery they are too. Michael seems to be conservative, having gone for a twin-carb. machine, while Paul once again has gone for performance with fuel-injected turbo beast. May they both enjoy their chariots without attracting too many cops, poles, trees or trouble. Remember the Commodore, boys.

It would seem that Warren has got himself an IBM look-alike and may be deserting the CP/M ranks for PC- or MS-DOS. In any event he seemed to be somewhat pre-occupied at the BBQ.

The boys organised the usual after-lunch cricket game and all those playing had a ball (or several). Paul swore that this was the last time he would organise it. We will see how he feels by the time of the next BBQ. Time heals most wounds, Paul.

Most people did leave their toys at home, but David brought his MAC* to use as a "very expensive terminal" for demonstrating a prototype Forth engine using the 65F11/12 CPU with built-in ROM Forth. Ray Gardiner lugged his Rabble, 2 monitors (1 RGB), and prototype colour hi-res board (intelligent) all the way from Shepparton, got it partly set up and found that he had left his boot and demo disks at home. End of that demo! We had to be satisfied with a description of all the wondrous things that this new Rabble expansion will be capable of early in 1985 (drool!).

On that note I will close for this year, leaving you all out there with the following thoughts:
* KAOS subs are now due
* OSI 8" disks are now available from me
* A HAPPY NEW YEAR TO EVERYBODY
* See you all in January 1985
Ralph Hess

KAOS W.A.

Our last meeting was a short one as only 3 members attended. So this is a reminder that our main meetings are always on the 3rd Sunday of each odd month. In future I will always place an add in the computer section of the Sunday Times, the week before each meeting.

I will not be at the next meeting but I hope to be able to attend the January KAOS meeting while I am on holidays in Melbourne.

Our next meeting is therefore on Sunday 2Øth January at 2.ØØpm in Guild House 56 Kishorn Rd, Mt Pleasant.
Gerry Ligtermoet,

SUPERBOARD

Newsletter of the Ohio Superboard User Group, 146 York Street, Nundah 4012.
©January, 1985.

Apologies are in order again regarding the Adventure program. I'll put it in when fully debugged. Future SUPERBOARD issues will be all software, unless someone else contributes something to these pages. Wherever it is possible, I will put programs in generic code to suit the various computer types in KAOS. Does anyone have any requests?

BINARY COUNTER

Here's a smart little two line program to keep your OSI busy for hours. A delay loop can be installed in line 20 to slow down the count so that you can see it at the fast end.

```
10 FORR=1 TO 32:PRINT:NEXT:R=49:S=53755:REM Binary Counter
20 SC=S+A:B=PEEK(SC)=R:POKE SC,R+B:A=B-B*A:GOTO 2Ø
```

For 32 x 64 mode, use SC=54207, or in 32 x 32 mode, SC=54239

BIORHYTHM CALCULATION

I don't believe the biorhythm theory. I've never seen any documented evidence to support it, and tests I've done myself have failed to show anything except random results. And I have done some tests! In 1980, I read somewhere that someone had used biorhythms to predict the poor performance of the favourite in the Melbourne cup. A lot of wild claims had been made about biorhythms, and the newspapers were full of stories of how some quite famous people had met untimely ends on their "critical days"

I decided to do my own research, and armed with a calculator, obtained the birth dates of top jockeys from the Who's Who of Racing, horses from sales catalogues, and birth and death dates of teenagers who had died on the roads. Unfortunately I didn't have a computer then, and not even a biorhythm calculator, so the whole thing was done by hand with the help of one of those 100 year calendars that were reproduced in the back of diaries.
All the statistical tests I applied showed results that were no different to random. If biorhythms do exist, then they have very little effect on our lives. Book and special calculator sellers were the only ones to benefit from that period of biorhythm madness. Here is the shortest biorhythm calculator program of all time! It only works accurately between March 1902 and February 2400, and if you don't type in the full year, you'll need to remember to add 100 for years 2000+, 200 for years 2100+ and etc. Somehow, I doubt if that will bother you much!

```
10 INPUT"Birth date";D,M,Y:PRINT:GOSUB4Ø:ND=C
20 INPUT"Destination date";D,M,Y,:PRINT:GOSUB4Ø:ND=ABS(C-ND)+1
30 PRINT"Days lived:";ND:PRINT:FORCD=23 TO 33STEP5:GOSUB5Ø:NEXT:END
40 C=INT(365.25*(Y+(M<3)))+INT(3Ø.6*(M+1-(M<3)*12))+D:RETURN
50 PRINTCD;"Day cycle:";INT((ND/CD-INT(ND/CD))*CD+.5):PRINT:RETURN
```

Not all computers in the club are OSI. If yours gives an answer of 1 to PRINT 1<3, change in line 40 to (Y-(M<3)) and (M+1+(M<3)*12).

— SUPERBOARD —

ROM BASIC UTILITIES

Mark Howell has produced a ROM version of the Extended Basic Commands tape to suit C1Ps and C1/4s using the DABUG monitor ROM. Renamed ROM BASIC UTILITIES, it occupies $E000 to $E7FF as a 2716, or can be ordered with a much improved Extended Monitor from $E800 to $EFFF in a 2732.

Mark has corrected a couple of small bugs he detected in the Auto, Copy, and Renumber routines, and in the space created by shifting the code around to make the machine code work in ROM, added a couple of extra commands. Unlike E.B.C. which conflicted with DABUG functions, R.B.U. is fully compatible with the monitor.

The main addition is a routine to provide a tape-to-tape copier, an altered version to the one I published in OSUG Newsletter #20 in pre-KAOS days. The program has a first in, first out buffer, and so provides a "computer original " reproduction of any programs on a tape, invaluable for use in copying a whole C60 cassette for a friend. Also, you can interrupt what you are presently doing to send a program over a modem to a friend without losing your present memory contents. You see what is going through on the screen.

All the other features remain as immediate mode commands which become part of the Basic language. Auto numbering, Renumbering, Delete, Trace, Copy line, Limited listing, and the terrific Change editor all make working with Basic programs a delight. For a review of E.B.C., see KAOS 3/1 page 5.

PRICES:	R.B.U. in a 2716	$10.80 including postage.
	R.B.U. + EXMON in a 2732	$18.80 " "

Mark Howell,

KAOS QUEENSLAND MEETING, 9th DEC 1984.

Attendance: 11 Computers: 5

The attendance was poor at our last meeting for the year. Obviously the advertising wasn't too successful and cost nearly $8, something we can no longer afford. BBCs outnumbered OSIs by one, and only the usual small band of devotees were in attendance.

Paul Brodie, John Froggatt, and Peter Meulman provided the BBCs. John was delighted to discover another 80 track disk user in Peter, who also had a colour monitor which turned out to be a star attraction for most of the meeting. The games had again grown and were even more complex than those available at the last meeting. A graphic adventure was spectacular, and an arcade adventure called Elite created an enormous universe which could be travelled by the player, as captain of a starship trader. Obviously based on the "Traveller" fantasy role playing games,goods could be traded, wars fought, and empires built.

Ian Mackenzie had a new catalog from Bison International of Apple clone fame. The Taiwanese copying industry is alive and well, and moving in the IBM direction. Alan Calvert is doing a course on computers and was considering the pros and cons of a Compsoft CP/M board. Other members were in the process of converting to disk.

The meeting closed at 5.30pm.

NEXT MONTH

Substring Search routine, and a good chance of that elusive adventure.
I'll take this opportunity to wish all KAOS members all the best for 1985.

Ed Richardson.

FORTH Part 7 THE R65F11
by Ray Gardiner

The nucleus layer in the Forth word set is an arbitary grouping of words with like characteristics. Usually these words are written in Assembler for the CPU concerned. However they may be coded in hi-level Forth if desirable.

NUCLEUS LAYER

```
!    *    */    */MOD  +  +!  -  /   /MOD  Ø<  Ø=  Ø>  1+  1-  2+  2-  2/  <  =  >
>R   ?DUP   @    C!   ABS  AND  C@  CMOVE  COUNT  D+  D<  DEPTH  DROP  DUP  I  J
MAX  MIN  MOD  NEGATE  DNEGATE  NOT  OR  OVER  PICK  R>  U<
```

Together with the Device layer, Interpreter layer, and the Compiler layer you have a complete self supporting Forth system. Sometimes the minimum run-time support system is called the KERNEL, this consists of the address interpreter and a minimum support package, usually the interpreter and a device driver support that may be required in a particular application. The KERNEL can be very compact, as small as 2-3K and can be placed in ROM with the other layers being added as required. Such a system is the ROCKWELL 65F11, 65F12 series of 65Ø2 based, single chip, complete Forth single chip computers.

THE ROCKWELL 65F11, 65F12

The 65F11 is a 4Ø pin dip package which contains a complete 65Ø2 CPU and 3K ROM containing the Forth KERNEL, 192 bytes of RAM, and interface circuitry. The interface circuitry includes two 16-bit programmable timer/counters, 16 bidirectional I/O lines (including four edge-sensitive lines and input latching on one 8-bit I/O port), a full duplex serial I/O channel, ten interrupts and an expandable bus structure.

The 65F11, with its two 8-bit ports is housed in a 4Ø-pin dip. For systems requiring additional I/O ports, the 64 pin QUIP version, the 65F12, provides a total of five 8-bit I/O ports.

The kernel of the high level Rockwell Single Chip Forth language in the preprogrammed ROM of the R65F11 and R65F12 is based on the popular fig-Forth model with extensions. All of the runtime functions are contained in the ROM, including 16 and 32 bit mathematical, logical and stack manipulations, plus memory and input/output operators. The RSC-Forth operating system allows an external user program written in Forth or Assembly language to be executed from external EPROM, or you may wish to use the 65FR1 development ROM, supplied on a 2764 by ROCKWELL.

The only external hardware required for a minimum system is a ROM containing your application program. And a crystal as well as power supply and connectors for the I/O lines and serial interface. During the program development phase you connect a serial terminal and then interactively develop the application program with the hardware connected to the I/O lines. This means that debugging and development proceeds interactively resulting in very fast project development times. When the development stage is complete the program can be programmed into EPROM on the board using the latching facility to hold address and data lines while each location is programmed.
Such a system will revolutionise the approach to a host of one-off applications such as industrial control and home computer projects.

NOTE: David Anear has completed a prototype PCB layout for both the R65F11 and R65F12. If enough KAOS members would like a printed circuit board perhaps a production run can be organised. The R65F11 chip costs $18.ØØ from Energy Control, P.O. Box 65Ø2, Goodna, Qld.

If you want a ready made system then you could contact New Micros Inc. 8Ø8 Dalworth, Grand Prairie, Texas 75Ø51, U.S.A. telephone (214) 642-5494. They sell both 65F11 and 65F12 boards in various configurations, for various prices starting from $US2ØØ or so.

Thanks to David for providing the information on the R65F11, I hope to present some design examples during the following months using the R65F11.E

WP65Ø2 V1.2 REVISITED

by John Whitehead

WP65Ø2 is my most used program, and along with my second most used program, the Assembler, is in separate EPROMs at the same address of $8ØØØ selected with a switch. Both run at $8ØØØ (not down loaded, with the exception of small sections of self modifying code between $Ø222 and $Ø2FF) and use workspace from $Ø3ØØ to the end of RAM.

Over the past year I have noted alterations I wanted to make to WP.

As my disassembled listing of WP was a bit tatty, I decided to make an assembler source code listing of WP. This was performed by using the symbolic disassembler writen in BASIC which converts M/code into an assembler source and puts it out on tape. This tape is then fed into the assembler. The lines containing data are then tidied up and comments added.

Half way through the task of adding comments , I ran out of RAM. I bought two more 6116 RAM chips and put them on a Tasker EPROM card I had and now have 32K of RAM.

Now after two months, I have a 32 page source listing of my Dabug compatible 48x12 EPROM version of WP65Ø2 V1.2. It contains comments of the M/code functions that I have found, the mods that I have made, and for sub-routines, it lists where the calls come from if there are less than six calls.

Finding the calls (JSR XXXX) was made easy, due to the search command in David Dodds Assembler in EPROM.

When I made my EPROM version of WP, I put the main core in the same place as it was in the tape version with just the individual bytes changed where needed. Code that was at $Ø222 to $ØFEF was relocated to $8222 to $8FEF which makes the listing compatible with both versions. Also when I modified code, I have not re-assembled it, just patched it in. This way I do not require another print-out of the whole listing. In the following mods, where it refers to $833C for example, use $Ø33C.

My source code can be fed into the Assembler and assembled if there is 32K of free RAM. An assembled listing can be feed into WP65Ø2 for printing out a bit at a time if there is not enough memory for assembly.

If you would like a copy of my source code for feeding into the Assembler or an assembled listing for feeding into WP, send me one blank C6Ø tape, money for return postage plus $1 and proof that you already have WP65Ø2 V1.2 (e.g. WP recorded in checksum on the tape you send) to 17 Frudal Cres., Knoxfield 318Ø. This listing could also be helpful to disk users of V1.2.

The following are the latest changes I have made to WP with the aid of the above source listing. The changes can be patched in as required. I have put mine in front and behind the main core. With the tape version they will need to go after the existing code and the 'start of text' pointer at $Ø241-2 changed to the end of the added code. Although most mods are small, it took a long time to find how to do it.

(1) When I first modified WP for Dabug and 48x12, I had to change some of the special characters to make it work. The line feed marker was CHR $7F (Dabug screen clear char) at $8228 and I had to change it to CHR $5B. Dabug 3 did not allow CHR $18 to CHR $1F to be used. Using $5B sometimes made G/EDIT difficult to read. With my modified Dabug 3J, I can use more characters and have changed the line feed char to $1E. This also required 'lowest char' at $8225 to be changed to $18.
If you have recorded text that has a different line feed character, it is possible to do a G/EDIT and change them all as:-
Press BREAK and change $ØØ25 to the line feed char used in the text, e.g. $5B.
Run WP at $8FØE. (Normal warm start at $ØØØØ jumps to $8FØB to reset the variables. By entering three bytes later, resetting is by-passed.)
Do a G/EDIT (without pressing 'return to menu' to any unused char e.g. ***
then press RETURN and do a second G/EDIT from *** to your line feed character.

(2) When using CTRL keys (1, M, X & B) with the shiftlock up, two characters appear in place of one. This is fixed by inserting STA $41 after STA $Ø217 at $8558. To do this replace the STA $Ø217 with JSR CTRLFIX and at CTRLFIX put STA $Ø217, STA $41 and RTS.

(3) When entering the L/EDIT mode, it allows FROM to be used to start editing anywhere in the text. I have altered the VIEW annd PRINT modes to also use FROM. I have also added a SIMULATE and HOLD mode that bypasses turning the printer on, to show where a page ends. (This is already in the disk version.)
Change the existing code between $87E9 and $87FA to BEQ $87F4, CMP #'S (SIMULATE and HOLD), BEQ $87F7, LDA #$6Ø, STA $Ø247, JSR TELETYPE ON (INC $Ø2Ø5 for normal printer), JMP VIEWP, NOP.
Add new code as:- VIEWP LDA $3A, STA $38, LDA $3Ø, STA $56, VIEWF, LDA $Ø247, PHA, LDA #$4C, STA $Ø247, JSR $82E1, JSR $8784, JSR $84E9, JSR $8335, JSR $82F4, PLA, STA $Ø247, JMP $87FB. Also change the jump at $87A1 to JMP VIEWF.

(4) I use a teletype model 35 as a printer. This uses paper on a roll without perforations. I have added code to print perforations consisting of a line of dashes at the beginning of the first page and at the end of every A4 page. Details of this can be found on my tape listing.

(5) It may sometimes be necessary to delete a large amount of text from an existing file. I have added a BLOCK DELETE that works the same as DELETE SENTENCE. You enclose the text to be deleted with a CTRL B and a CTRL X, use the BLOCK VIEW to check the text to be deleted, then go to L/EDIT, put the cursor under the CTRL B character and press DB.
The new code for this is:- BLOKX CMP #'B, BEQ DELB, LDA $47, JMP $8CØ6, DELB LDX $26, JMP $8C99
Change at $8C8F to NOP, NOP, JMP BLOKX

(6) My last mod was to alter the ZAP command so that the whole word ZAP had to be entered for the file to be deleted. As stated at the beginning, Warm Start is at $8FØB, this sets up variables and prints the menu, then waits for a key press at $8F3F. Below this are all the compares for the mode required. The new code needed is:- GZAP JSR $83FF, CMP #'A, BNE $8F9D, (this branch has to point to JMP $8465. It may not be at $8F9D), JSR $8332, JMP $ØØØØ.
Existing code to change is at $8F45 as:- CPX #'Z, BEQ GZAP, JSR $82E1, NOP, NOP, NOP, and at $8F63 as:- CPX #'V, BNE $8F73, JMP $8795

(7) There is another fault with WP that I have not been able to sort out yet, and that is to do with workspace full. This is what I have found out so far:- ZAP puts a @ at the start of workspace. TYPE checks memory and fills it with $FB from the first @ to end of RAM. If the top of RAM is $1FFF, workspace top is set to $1EFF and stored in $1Ø and $5B. The last 256 bytes are used for line and global editing. When text is entered and the characters get to $1EFE, TYPE shows Ø bytes free and $1EFF contains @. If one more character is entered it shows 65535 bytes free. More text can be entered until it reaches $1FFE when OM?> shows and this gives 6528Ø bytes free. Once the bytes free are incorrect, line and global editing does not work correctly and can delete all your text.

For those of you who use WP65Ø2 V1.2 or K1.2 and are not too familiar with M/Code, have a go at one of the above mods. As long as you keep the original tape, no harm can be done if you make a mistake. You will need a mnemonic to hex conversion chart and the Extended Monitor to check the modified code. The M/Code above contains labels. These are swapped for the address where you put the new code, e.g. GZAP in (6) could be $1ØØ3.

It is not possible to have Exmon at the top of RAM and protect it from WP as can be done with BASIC as WP fills all unused RAM with $FB's nearly every time return is pressed. Exmon can be in write protected RAM, EPROM or in a section of RAM that is not continuous from WP's workspace. It can be below WP's workspace and the start of text pointer at $8241-2 set to the end of Exmon.

FOR SALE

Microline 83A 15 inch printer, serial and parallel interface, removable tractor feed. $6ØØ.ØØ
Warren Schaeche

C1P converted to C4P, has Tasan video board and Rabble expansion board,
2 x 5.25" drives, 48K RAM, lots of disks. $6ØØ.ØØ
Warren Schaeche

C4P series 2 computer (standard), 2 joysticks, cables, modified B/W 14" TV, manuals, 2Ø cassettes of games, including Space Invaders, High Noon, Othello. Hardware is in good working order. $3ØØ inc. Courier costs
M. Black,

Monitor, converted H.M.V. 17in. TV with direct video input and green filtered screen. $4Ø.ØØ
Colin Haustorfer

Series II Superboard in KAOS case, 8K RAM and DABUG III monitor. Sanyo 12" TV converted for video. $19Ø.ØØ
Bill Grimes

GRAFIX SEB-1 graphics expansion board for the C1P, the board has not been filled but I have all the required chips and manual. Will accept reasonable offer.
C. Bear

Ohio Scientific C1P series 2 computer with power supply, several games on tape, documentation, User Group newsletters, very good condition. $15Ø.ØØ ono
Andrew

Registered by Australia Post
Publication No. VBG4212

If undeliverable return to
KAOS, 10 Forbes St
Essendon, Victoria 3040